Improved regularization in extreme learning machines

Andrea Carolina Peres Kulaif
LBiC/DCA/FEEC
University of Campinas (Unicamp)
Campinas, SP – Brazil
E-mail: deiacpk@dca.fee.unicamp.br

Fernando J. Von Zuben
LBiC/DCA/FEEC
University of Campinas (Unicamp)
Campinas, SP – Brazil
E-mail: vonzuben@dca.fee.unicamp.br

***Abstract*—Extreme learning machines (ELMs) are one-hidden layer feedforward neural networks designed to be trained at a low computational cost and to exhibit a direct control of the generalization capability, both for regression and classification tasks. By means of ridge regression, it is possible to properly control the norm of the connection weights at the output layer. Restricted to regression tasks and MLP-like topologies, the main contribution of this paper is to indicate that a more refined search for the parameter of the ridge regression, based on a golden section mechanism, consistently guide to better performance in terms of generalization, when compared to what is done in the state-of-the-art proposals for ELMs in the literature, and no matter the number of neurons at the hidden layer. The improvement in regularization comes at an additional computational cost, though we still are dealing solely with the adjustment of the connection weights at the output layer, which implies that the effective cost remains orders of magnitude below what we obtain when training an MLP neural network.**

Keywords—Extreme learning machines; golden section search; regression problems; generalization capability.

I. INTRODUCTION

Feedforward neural networks, in general, are universal approximators, such as multilayer perceptron (MLP) [6] and radial basis function (RBF) [14] neural networks. The universal approximation capability fully supports the existence of a neural network that complies with a given threshold for the training error. However, this is an existential property, and we still do not have a systematic and computationally effective way of properly and automatically defining the required number of neurons and setting the connection weights so that the resulting neural network maximizes the generalization capability. In other words, achieving a given threshold for the training error is ease to attain in practice, and the real challenge is to control the generalization capability of the neural network.

Even with the continued increasing in the availability of computational resources, an additional drawback of MLP and RBF neural networks is associated with the necessity of properly setting the weights at the hidden layer. In the MLP case, a nonlinear optimization problem should be solved. In the RBF case, self-organizing methods are generally applied to set the center of the radial basis functions. Those are iterative and time consuming procedures, besides being subject to local optimal solutions.

This work has been supported by grants from Capes and CNPq.

Support vector machines (SVMs) [17] are founded on statistical learning theory and come to provide better solutions to both the automatic definition of the number of neurons at the hidden layer and of the connection weights. The number of neurons is associated with the number of support vectors and the connection weights are obtained by the definition of a hyperplane of maximum margin in the feature space, which is a convex optimization problem. Though supported by theoretical results guiding to the minimization of the structural risk [15], the practical performance of SVMs still depends on a proper choice of the kernel function and its corresponding parameter(s) [12]. Besides, the computational cost to solve the resulting quadratic optimization problem subject to equality or inequality constraints [16] is still a practical drawback of SVMs.

Extreme learning machines (ELMs) [9][11] were then proposed to circumvent those practical limitations of the aforementioned approaches, more specifically: (1) ELMs training phase is orders of magnitude less computationally intensive, when compared to what is required for MLPs, RBF neural networks, and SVMs; (2) The optimization problem is convex, which means that there is no local minima, and there is no kernel function to be defined; (3) The generalization capability is directly controlled by setting a single regularization parameter of a ridge regression problem; (4) Though defined randomly, the number of hidden neurons and the connection weights, when defined obeying predefined intervals of admissible values, do not influence significantly in the generalization capability; (5) ELMs have been demonstrated to be generalized versions of MLPs, RBF neural networks and SVMs [10], with similar performance in terms of generalization.

Those amazing properties of ELMs may be supported by two initiatives: (I-1) the mapping of the original problem to a feature space of much higher dimension when compared to the dimension of the input space, even without the use of a kernel that obeys the Mercer condition [15], as done in SVMs; (I-2) the direct exploration of an empirical result of Bartlett [1][2], which asserts that the size of the weight vector at the output layer is more relevant for the generalization capability than the configuration of the neural network, in terms of number of neurons and format of the activation function.

Since it was originally proposed [11], and given that the computational cost to train ELMs is very attractive, there have been several attempts to improve initiatives I-1 and I-2,

looking for gain in performance, possibly at the price of an additional computational cost. Concerning initiative I-1, there are several alternative proposals in the literature aiming at converting the training phase to an incremental procedure, with a sequential introduction of neurons at the hidden layer. They are called incremental extreme learning machines (I-ELMs) and some relevant contributions are [7], [8] and [18].

The main contribution of this paper is associated with initiative I-2 and is restricted to regression problems and MLP-like topologies, even though the extensions to classification problems and other types of ELM's topologies are straightforward. We concentrate ourselves at the ridge regression problem associated with the definition of the connection weights at the output layer. Our empirical results considering seven benchmarks from the literature, clearly indicates that it is consistently worthwhile to refine the search for the regularization parameter in the ridge regression problem. Until now, the more advanced result, presented in [10], is associated with a rather crude search (discretely sampling an interval of admissible values) for a single value for the regularization parameter of the ridge regression problem, using a validation dataset to guide the search. Here we start from this sampling procedure and introduce an additional golden section search to deeply refine the value of the regularization parameter. Besides, we are no more looking for a single value for the regularization parameter for each regression task, as done in [10]. In fact, once defined the number of hidden neurons and the regression task, the random configuration of the connection weights at the hidden layer strongly influences the optimal choice of the regularization parameter.

The paper is organized as follows: Section II briefly formalizes extreme learning machines, with emphasis on the formulation and solution of the ridge regression problem associated with the training phase. Section III explains the motivation and fundamental aspects of the refined search we are proposing here. Section IV presents the benchmarks and the obtained results, followed by a comparative analysis with the state-of-the-art ELM and also with conventional MLP. Section V outlines some concluding remarks and some perspectives for the further steps of the research.

II. Extreme Learning Machines

A. Preliminaries

Let us consider the one-hidden layer neural network presented in Figure 1.

The input-output mapping performed by this neural network is given in equation (1), as follows:

$$\hat{s}_{lk} = \sum_{j=1}^{n} w_{kj} f\left(\sum_{i=1}^{m} v_{ji} x_{li} + v_{j0}\right) + w_{k0} \tag{1}$$

In equation (1), $k \in \{1,\ldots,r\}$ is the index of the output and $l \in \{1,\ldots,N\}$ is the index of the input-output pattern belonging to the training dataset. Notice the existence of a fixed input equal to 1 associated with each neuron in the network.

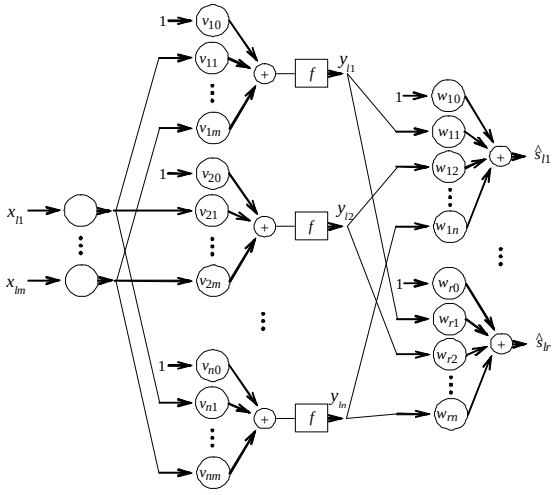

Fig. 1. One-hidden layer neural network with *m* inputs, *n* hidden neurons, and *r* outputs.

The activation of each hidden neuron $j \in \{1,\ldots,n\}$ for each input pattern $l \in \{1,\ldots,N\}$ is going to be represented as:

$$h_{lj} = f\left(\sum_{i=1}^{m} v_{ji} x_{li} + v_{j0}\right) \tag{2}$$

Now it is possible to express, in a single matrix H, the activation of all neurons at the hidden layer for all the input patterns, producing:

$$H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Nn} \end{bmatrix} \tag{3}$$

Denoting $\mathbf{w}_k = \begin{bmatrix} w_{k0} & w_{k1} & \cdots & w_{kn} \end{bmatrix}^T$, and inserting a column of 1's as the first column of H, the k-th output $\hat{\mathbf{s}}_k$ ($k \in \{1,\ldots,r\}$) of the neural network in Figure 1, for all the input patterns, is then provided by:

$$\hat{\mathbf{s}}_k = \begin{bmatrix} \hat{s}_{1k} \\ \hat{s}_{2k} \\ \vdots \\ \hat{s}_{Nk} \end{bmatrix} = \begin{bmatrix} 1 & h_{11} & h_{12} & \cdots & h_{1n} \\ 1 & h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & h_{N1} & h_{N2} & \cdots & h_{Nn} \end{bmatrix} \begin{bmatrix} w_{k0} \\ w_{k1} \\ \vdots \\ w_{kn} \end{bmatrix} \Rightarrow \hat{\mathbf{s}}_k = H\mathbf{w}_k \tag{4}$$

Notice that, even if you change the activation function $f(\cdot)$, thus producing distinct neural network models, such as MLP, RBF and even SVM [10], the algebraic structure of equations (1) to (4) remains the same.

B. The regularized optimization problem: ridge regression

In ELMs, solely the weights at the output layer are adjustable, with the weights between the input and hidden layer being randomly defined. The number n of hidden neurons may also be defined a priori.

For each output k, $k \in \{1,\ldots,r\}$, to obtain the weights at the output layer, the available dataset is divided into training

and validation, producing the training dataset $\{\mathbf{x}_l, s_{lk}\}_{l=1}^{N}$ and the validation dataset $\{\mathbf{x}_l^{(v)}, s_{lk}^{(v)}\}_{l=1}^{N_v}$. It is then possible to define $\mathbf{s}_k = [s_{1k} \quad s_{2k} \quad \cdots \quad s_{Nk}]^T$ and $\mathbf{s}_k^{(v)} = [s_{1k}^{(v)} \quad s_{2k}^{(v)} \quad \cdots \quad s_{N_v k}^{(v)}]^T$.

The weights at the output layer of the *k*-th neuron are the solution of the following regularized optimization problem:

$$\mathbf{w}_k^* = \arg \min_{\mathbf{w}_k \in \Re^{n+1}} \|\mathbf{w}_k\|^2 + C \times \|H\mathbf{w}_k - \mathbf{s}_k\|^2 . \quad (5)$$

Given the value of C and supposing $N > (n+1)$, the solution is expressed by:

$$\mathbf{w}_k^* = \left(\frac{I}{C} + H^T H\right)^{-1} H^T \mathbf{s}_k . \quad (6)$$

The validation dataset $\{\mathbf{x}_l^{(v)}, s_{lk}^{(v)}\}_{l=1}^{N_v}$ should be employed to properly set the value of the regularization parameter C.

C. Definition of the regularization parameter

In the optimization problem in (5), when we set $C \to \infty$, we obtain the solution without regularization, where only the error at the output is taken into account. In this case, $\|\mathbf{w}_k\|$ is unrestricted.

On the other hand, for $C > 0$ and finite, there is a compromise between the error at the output and the norm of $\mathbf{w}_k$. To achieve a proper compromise, Huang *et al.* [10] adopted the value of C in the set $\{2^{-24}, 2^{-23}, \cdots, 2^{+24}, 2^{+25}\}$, so that the choice is the one that minimizes the error for the validation dataset $\{\mathbf{x}_l^{(v)}, s_{lk}^{(v)}\}_{l=1}^{N_v}$. Notice that, in [10], a single C is defined for each regression task.

III. The Proposed Methodology

We are going to propose a more refined search strategy to define the regularization parameter C. The motivation is twofold, and has been supported by practical results obtained when dealing with regression problems:

(1) Small deviations in C may guide to significant variation in the solution;

(2) For any reduced search interval with candidate values for C, the curve associating the value of C with the validation error tends to exhibit a quasi-convex behavior.

The use of a unidimensional search in the whole interval $(0,+\infty)$ is not recommended, because the quasi-convexity of the curve associating the value of C with the validation error may be violated. Given that the quasiconvexity property of the aforementioned curve is a necessary condition for the application of a unidimensional search, we have decided to consider the search procedure of Huang *et al.* [10] and, after the definition of the best choice for C in the set $\{2^{-24}, 2^{-23}, \cdots, 2^{+24}, 2^{+25}\}$, we perform a golden section search [5] around this value, taken the right and left neighbor values of C in the set as extreme points of the refined search interval.

Besides, given that for each regression task 50 repetitions are going to be performed (see Section IV), we are going to proposed one distinct C for each ELM, and not the same C for all the ELMs associated with that specific regression task.

Though more computationally intensive, this new procedure conceived to properly optimize the regularization parameter C will be shown to exhibit a consistent better performance, thus contributing to improve the generalization capability of the obtained ELMs.

IV. Experimental Results

This section presents the experiments and results obtained with the proposed methodology of Section III, as well as comparisons with the state-of-the-art ELM [10] and standard MLP, trained with a second-order nonlinear optimization algorithm [3]. The stopping criterion for the MLP will be the minimization of the error for the validation dataset or 1,500 epochs of training.

A. Benchmark Datasets

All datasets considered here are associated with regression problems, and have already been considered as benchmarks in [10]. However, we adopted here a distinct partition, containing training (60%), validation (20%), and test (20%) datasets.

The datasets present distinct characteristics, including number of samples and number of features (dimension of the input space). Table I specifies the main aspects of the datasets. The datasets Bodyfat, Space-ga, Quake, and Strike can be found in [13], and Abalone, Pyrim and Housing are available in [4]. It was made a scaling for the input data in the interval [–1,+1], and for the output data in the interval [0,+1].

TABLE I – Attributes of the Datasets

Datasets	Number of Samples	Features
Pyrim	74	27
Bodyfat	252	14
Housing	506	13
Strike	625	6
Quake	2178	3
Space-ga	3107	6
Abalone	4177	8

B. Parameters

The simulations were performed in MATLAB 7.11.0.584 - 64-bit, running on Intel Core i5-2430M, 2.4GHz with 4GB of RAM. The results obtained when employing the methodology proposed by Huang *et al.* [10] will be presented here for comparison. As explained at the end of Section II, in this case the value of C will be given by one of the elements of the set $\{2^{-24}, 2^{-23}, \cdots, 2^{+24}, 2^{+25}\}$. Table II presents the obtained values for each dataset. They are not the same results presented in [10], because there the weights of the hidden layer were initialized randomly in the range [–1,+1], the number of hidden neurons was fixed in 1,000, and there was no test dataset there (only training and validation datasets were considered).

TABLE II – Obtained *C* using the methodology of Huang et al. [10] for each dataset

Datasets	*C*
Pyrim	2^0
Bodyfat	2^7
Housing	2^2
Strike	2^2
Quake	2^0
Space-ga	2^7
Abalone	2^8

In the golden section search implemented to support our proposal of Section III, the threshold for the search is the minimum between 10^{-5} and 0.01% of the initial search interval. In our experiments, the number of hidden neurons was fixed in 100 and the weights of the hidden layer were initialized randomly in the range [−0.5,+0.5]. Those values differ from the ones adopted in [10], because they produced more stable numerical results. Besides the holdout approach for validation of the obtained regression models, we will also employ here the k-fold cross-validation approach, not considered in [10].

TABLE III – Simulation Results (Holdout)
ELM with fixed *C* (see Table II) – 100 hidden neurons

Datasets	RMSE Valid.	Dev. Valid.	RMSE Test	Dev. Test	Traininig Time(s)
Pyrim	0.1112	0.0372	0.1152	0.0394	0.0162
Bodyfat	0.0296	0.0122	0.0340	0.0127	0.0197
Housing	0.0857	0.0118	0.0854	0.0136	0.0268
Strike	0.2735	0.0119	0.2735	0.0118	0.0240
Quake	0.1730	0.0077	0.1724	0.0071	0.0636
Space-ga	0.1814	0.1151	0.1756	0.0859	0.0908
Abalone	0.0756	0.0027	0.0770	0.0029	0.1221

C. Simulations Results – Part I

All results will be presented after 50 runs, so that it will be exhibited root mean squared error (RMSE), standard deviation (Dev.), and training time. Before each run, the training, validation, and test datasets are reshuffled and are the same for all methods.

Table III shows the results for the simulations with *C* fixed in the values of Table II. Table IV shows the obtained results, with holdout, for MLP and for the proposal of this paper, with a refined search for *C* (denoted Variable *C*).

Even though the training time is a bit higher when compared to the results in Table III, the gain in performance (validation error) is significant to justify the application of the golden section search. There is improvement in all cases and the computational cost is still much lower than the one produced by the MLP training, with a competitive generalization performance (except for the Strike dataset).

In the k-fold partitioning, the search is performed based on the mean squared error of k folders, finding the optimal *C* as the one with the best average performance considering each one of the *k* folders as the validation dataset. The results of these experiments are presented in Table V. Here, there is no comparison with the results in [10], because their experiments are restricted to the holdout approach only.

The training time using k-fold cross-validation is higher when compared to the holdout case. However, the results are more reliable and the improvement in accuracy is noticeable. We applied the Wilcoxon Signed-Rank Test in order to check for statistical difference in accuracy, when performing the golden section search for the parameter *C*, compared to MLP and the proposal in [10].

TABLE IV – Simulation Results (Holdout): MLP and Variable *C* – 100 hidden neurons

Datasets	MLP					ELM – Variable C				
	RMSE Valid.	Dev. Valid.	RMSE Test	Dev. Test	Training Time(s)	RMSE Valid.	Dev. Valid.	RMSE Test	Dev. Test	Training Time(s)
Pyrim	0.1104	0.0321	0.1438	0.0537	3.5131	0.1030	0.0338	0.1194	0.0426	0.0955
Bodyfat	0.0245	0.0116	0.0321	0.0133	12.8177	0.0248	0.0125	0.0306	0.0141	0.2346
Housing	0.0740	0.0121	0.0809	0.0124	24.8912	0.0818	0.0117	0.0843	0.0145	0.3051
Strike	0.1451	0.0198	0.1705	0.0384	29.8012	0.2504	0.0176	0.2978	0.1283	0.3747
Quake	0.1719	0.0078	0.1733	0.0070	103.3229	0.1725	0.0077	0.1732	0.0072	0.8964
Space-ga	0.1326	0.0045	0.1336	0.0048	147.3941	0.1367	0.0048	0.1478	0.0304	1.2787
Abalone	0.0748	0.0026	0.0766	0.0025	198.1542	0.0750	0.0026	0.0770	0.0033	1.7191

TABLE V – Simulation Results (k-fold): MLP and Variable *C* – 100 hidden neurons

Datasets	MLP					ELM – Variable C				
	RMSE Valid.	Dev. Valid.	RMSE Test	Dev. Test	Training Time(s)	RMSE Valid.	Dev. Valid.	RMSE Test	Dev. Test	Training Time(s)
Pyrim	0.0872	0.0129	0.1400	0.0298	5.6699	0.0749	0.0222	0.1093	0.0372	2.3481
Bodyfat	0.0217	0.0055	0.0359	0.0132	19.3082	0.0187	0.0076	0.0318	0.0151	7.9962
Housing	0.0671	0.0029	0.0784	0.0111	38.7696	0.0726	0.0102	0.0800	0.0129	16.0558
Strike	0.1283	0.0064	0.1665	0.0456	31.3439	0.2338	0.0186	0.2783	0.0596	19.8318
Quake	0.1698	0.0017	0.1727	0.0070	117.3793	0.1656	0.0074	0.1724	0.0070	37.2677
Space-ga	0.1313	0.0012	0.1346	0.0041	238.0573	0.1330	0.0059	0.1397	0.0058	98.5877
Abalone	0.0743	0.0008	0.0761	0.0028	320.0404	0.0736	0.0029	0.0764	0.0031	132.5398

See Table VI for details. Signals + means that the method proposed here is better. Signals ~ and – mean that the results are statistically equal or lower in performance, respectively. The p-value appears in parenthesis.

It is well known that ELMs are able to provide competitive results when compared with MLPs, in terms of generalization. Nonetheless, it is remarkable the gain in the computational burden. With the results of this paper, we confirm these results and go further, providing a clear evidence that a refined search for C can guide to even better results in terms of generalization for ELMs, when compared with the state-of-the-art in [10]. Notice that the optimal values of C we have obtained are distinct from the ones in Table II.

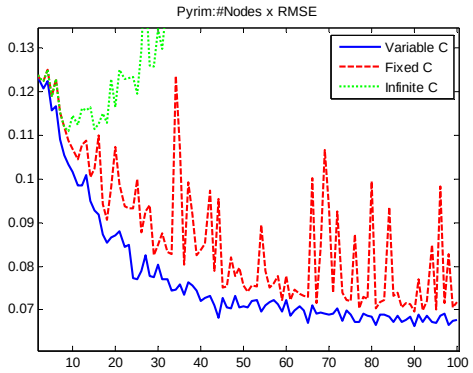

Fig. 2. Pyrim: Comparison among Infinite C, Fixed C=2^0 and Variable C.

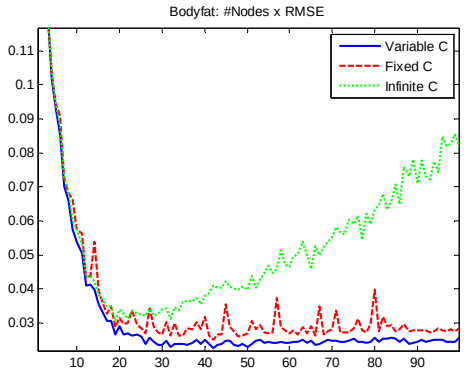

Fig. 3. Bodyfat: Comparison among Infinite C, Fixed C=2^7 and Variable C.

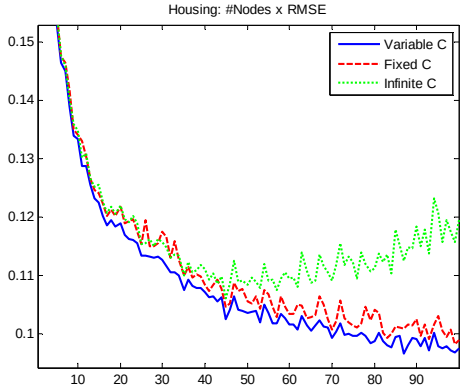

Fig. 4. Housing: Comparison among Infinite C, Fixed C=2^2 and Variable C.

D. *Simulations Results – Part II*

Now we are going to indicate that a proper choice of C is necessary when we vary the number of hidden neurons from 1 to 100. Besides the C produced by the golden section search (called here **Variable *C***), we present the performance with **Infinite *C*** (absence of regularization), and with the **Fixed *C*** methodology proposed in [10], using the values in Table II.

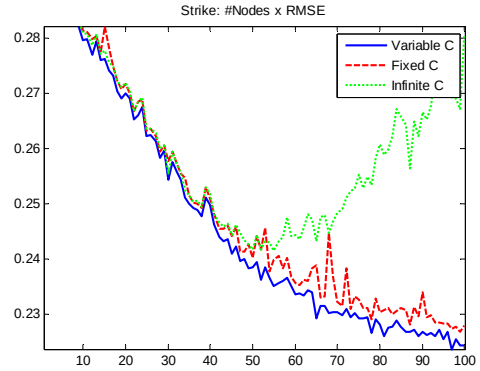

Fig. 5. Strike: Comparison among Infinite C, Fixed C=2^2 and Variable C.

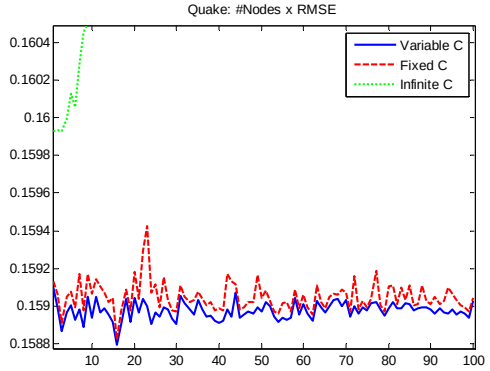

Fig. 6. Quake: Comparison among Infinite C, Fixed C=2^0 and Variable C.

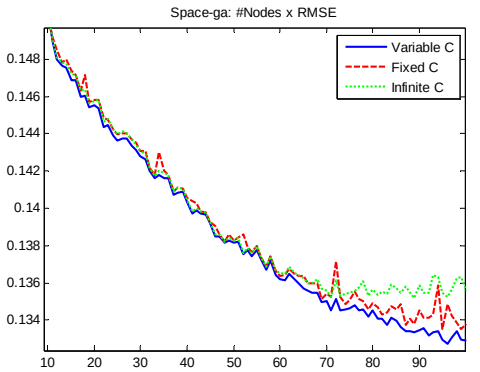

Fig. 7. Space-ga: Comparison among Infinite C, Fixed C=2^7 and Variable C.

Figures 2 to 8 show the root mean squared error (RMSE), considering the test dataset, of 50 runs with the holdout approach, again with the same datasets for the three curves. The use of variable C (one specific value of C for each number of hidden neurons) guides to a consistent superior performance.

TABLE VI – Wilcoxon Signed-Rank Test

	Pyrim	Bodyfat	Housing	Strike	Quake	Space-ga	Abalone
ELM + Holdout: Variable *C* versus Fixed *C*	+ (1.107×10^{-9})	+ (5.172×10^{-9})	+ (1.628×10^{-9})	+ (1.630×10^{-9})	+ (1.097×10^{-8})	+ (2.395×10^{-9})	+ (1.101×10^{-7})
Holdout: ELM Variable *C* *versus* MLP	~ (0.093)	~ (0.908)	– (4.919×10^{-7})	– (7.555×10^{-10})	~ (0.611)	– (7.541×10^{-10})	– (0.016)
k-fold cross-validation: ELM Variable *C* *versus* MLP	+ (0.002)	+ (0.006)	– (6.667×10^{-4})	– (7.553×10^{-10})	+ (0.002)	– (0.025)	~ (0.674)

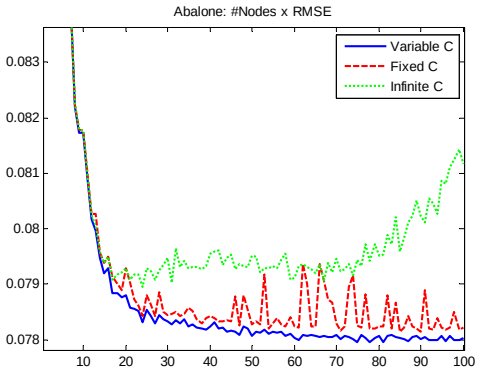

Fig. 8. Abalone: Comparison among Infinite *C*, Fixed $C=2^8$ and Variable *C*.

V. Concluding Remarks

The more relevant aspects that make ELMs so attractive for application in regression and classification problems are: (A1) their extremely low computational cost for training; (A2) the easiness with which we access the regularization parameter; (A3) the possibility of employing a very broad range of activation functions for the neurons at the hidden layer; (A4) the robustness of the performance when we vary the number of hidden neurons and the interval to set hidden layer weights.

This paper explores all those four aspects to propose an ELM with improved generalization capability. Given aspect A1, it is reasonable to propose a more demanding methodology which still maintains the computational cost at low levels. Aspect A2 is fundamental to our methodology, once we directly manipulate this parameter. Concerning aspects A3 and A4, we have adopted hyperbolic tangent as the activation function, and we have arbitrarily fixed (according to what has been adopted in the literature) the number of hidden neurons and the interval for the uniform random generation of the hidden layer weights.

Basically, we add a golden section unidimensional search to refine the choice of the regularization parameter when defining the connection weights at the output layer. The additional computational cost associated with our proposal is fully supported by the consistent gain in performance achieved.

Here the results are restricted to regression problems. However, it is straightforward to adapt the same methodology to deal with classification problems, and this extension will be made as the next step of the research. We are also working on the adaptation of the proposed methodology to deal with incremental ELMs.

References

[1] Bartlett, P.L. For valid generalization the size of the weights is more important than the size of the network. Advances in Neural Information Processing Systems, vol. 9, pp. 134-140, 1997.

[2] Bartlett, P.L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Transactions on Information Theory, vol. 44, no. 2, pp. 525-536,1998.

[3] Battiti, R. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. Neural Computation, vol. 4, no. 2, pp. 141-166, 1992.

[4] Blake, C.L., Merz, J.C. UCI Repository of Machine Learning Databases, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] Gerald, C.F., Wheatley, P.O. Applied Numerical Analysis. Seventh Edition, Addison-Wesley, 2004.

[6] Hornik, K., Stinchcombe, M., White, H. Multi-layer feedforward networks are universal approximators. Neural Networks, vol. 2, no. 5, pp. 359-366, 1989.

[7] Huang, G.-B., Chen, L. Enhanced random search based incremental extreme learning machine. Neurocomputing, vol. 71, pp. 3460-3468, 2008.

[8] Huang, G.-B., Chen, L., Siew, C.-K. Universal Approximation Using Incremental Constructive Feedfoward Networks with Random Hidden Nodes. IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879-892, 2006.

[9] Huang, G.-B., Wang, D.H., Lan, Y. Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics, vol. 2, pp. 107-122, 2011.

[10] Huang, G.-B., Zhou, H., Ding, X., Zhang, R. Extreme Learning Machines for Regression and Multiclass Classification. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, vol. 42, no. 2, pp. 513-529, 2012.

[11] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN'2004), vol. 2, pp. 985-990, 2004.

[12] Lima, C.A.M., Coelho, A.L.V., Von Zuben, F.J. Hybridizing mixtures of experts with support vector machines: Investigation into nonlinear dynamic systems identification. Information Sciences, vol. 177, pp. 2049-2074, 2007.

[13] Mike, M. Statistical Datasets, Dept. Statist., Univ. Carnegie Mellon, 1989. [Online]. Available: http//lib.stat.cmu.edu/datasets/

[14] Park, J., Sandberg, I.W. Universal approximation using radial-basis-function networks. Neural Comp., vol. 3, no. 2, pp. 246-257, 1991.

[15] Schölkopf, B., Smola, A.J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2001.

[16] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J. Least Squares Support Vector Machines, World Scientific Publishers, 2002.

[17] Vapnik V.N. The Nature of Statistical Learning Theory, 2nd edition, Springer, 1999.

[18] Yang, Y., Wang, Y., Yuan, X. Bidirectional Extreme Learning Machine for Regression Problem and Its Learning Effectiveness, IEEE Transactions on Neural Networks, vol. 23, no. 9, pp. 1498-1505, 2012.